# Práctica 4
## Hashing

## Ejercicio 1

Dados los enteros 5, 20, 3, 1000, 45, 27, 25, una tabla hash de tamaño 11 y una estrategia de resolución de colisiones de dispersión abierta, con la siguiente función de dispersión que retorna una posible posición para el elemento "x":

```
public int h(int x){
        return x%11;
}
```

Graficar el estado de la tabla antes y después de insertar los elementos.

## Ejercicio 2

Dados los enteros 5, 20, 3, 1000, 45, 27, 38, una tabla hash de tamaño 11 y una estrategia de resolución de colisiones de dispersión cerrada, con la siguiente función de dispersión que retorna una posible posición para el elemento "x":

```
public int h(int x){
        return x%11;
}
```

Graficar el estado de la tabla antes y después de insertar los elementos, bajo las siguientes resoluciones de colisiones:

a. Resolución de colisiones lineal: $f(i) = i$ → Función de disperción: $h_i(x)= (h(x) + f(i)) \% 11$

b. Resolución de colisiones cuadrática: $f(i) = i^2$ → Función de disperción: $h_i(x)= (h(x) + f(i)) \% 11$

## Ejercicio 3

Dados los enteros 5, 20, 3, 1000, 45, 27, 25, 67, 105, 3, 36, 39 dos tablas de hash de tamaño 11, y una estrategia de resolución de colisiones de dispersión del cuco, con las siguiente funciones de dispersión que retornan posibles posiciones para el elemento "x":

- h1(key) = key%11
- h2(key) = (key/11)%11

Graficar el estado de cada tabla antes y después de insertar los elementos.

## Ejercicio 4

Una contraseña utiliza información secreta para controlar el acceso a determinados recursos. Los sistemas seguros almacenan sus contraseñas de tal forma que si obtiene esta información, no pueda inferir las contraseñas de los usuarios a partir de la misma.
Usted debe implementar un sistema de validación de contraseñas como el descripto anteriormente utilizando una tabla de dispersión abierta de tamaño 23. La clave de la tabla de dispersión se debe calcular a partir de tanto el nombre de usuario como de la contraseña del mismo. En la posición de la tabla de hash indicada por la clave obtenida previamente, se almacenan los nombres de los usuarios que usan esa posición de la tabla hash.

Suponiendo que se usa como función de dispersión para mapear contraseñas a entradas en la tabla de dispersión:

```
public int getHashEntry(string user, string passwd){
        int hash = 5381;
        foreach (char c in user)
                hash = (hash * 7) + (int) c;

        foreach  (char c in passwd)
                hash = (hash * 7) + (int) c;
```

```
        return hash % 23;
    }
```

(a) Implemente el sistema de verificación de contraseñas, definiendo los siguientes métodos:

- guardarClave(usuario, passwd) Calcula la entrada en la tabla de hash y almacena en la misma el nombre del usuario
- verificarClave(usuario, passwd):boolean Retorna verdadero si se verifica que el usuario tiene esa clave. Retorna falso en caso contrario.